Focus Windows: A Tool for Automated Provers*

Florina Piroi

Research Institute For Symbolic Computation,
4232 Hagenberg, Austria
Florina.Piroi@risc.uni-linz.ac.at

1 Introduction

Taking a look at mathematical publications we see that the proofs presented there are in linear form. Some may be quite long, spreading over several pages. Therefore, readers of mathematical publications may sometimes find it inconvenient to turn pages in order to look up some formula whose label appears at the point of reading. This lookingup operation has an negative influence on the understanding of the proofs, even if their presentation in linear form is nicely structured and well presented.

The design and implementation of a theorem proving system is a challenging task. But most automated theorem provers do not put emphasis on producing proofs that are easy to read and understand [9]. Even when they do not present the proofs in a linear form but in a tree-like form [6], one may still have to jump to various formulae in the tree for being able to check or understand the validity of a particular step.

Focus windows were first introduced as a technique for proof presentation in [2]. It can be applied to both human and machine produced proofs. But whereas a human reader is likely to apply it in his mind (sometimes with the help of pen and paper), an automated prover can make use of this method by using an algorithm that processes its output and presents it to the user in an appropriate form. One should notice that this method is *not* a prover. The present paper describes the implementation and the use of the focus windows technique in the frame of the *Theorema* system [3].

One of the main concerns in the *Theorema* system is the emphasis on attractive proof presentation, making proofs easy to read and interact with. The inferences try to imitate the natural steps used by human provers. Consequently, the system is designed to produce proofs that resemble the ones generated by humans, containing formulae and explanatory text in English. Hyperlinks to the formulae referenced (presenting them in a small auxiliary window), nested cell brackets that allow contracting sub-proofs to one line, color codes that distinguish (temporary) proof goals from formulae in the (temporary) knowledge base are other tools that help the user to read and browse the proofs. Still, reading

* partially supported by the RISC PhD scholarship program of the government of Upper Austria and by the FWF (Austrian Science Foundation) SFB project P1302

and understanding long linear proofs is difficult even for proofs generated by the typical *Theorema* provers [7], [2].

By focus windows we provide a tool to overcome this difficulty. The idea of this technique is to analyze and collect exactly those formulae that are used and produced at each proof step (we call these formulae the formulae *relevant* for this proof step), composing then a window that contains exactly these formulae [7], [2]. While implementing and exploring this technique, we extended this tool for the use of the developers of the *Theorema* system, who can inspect not only the formulae used and produced at a particular proof step but also other information that helps them in inspecting and improving the provers of the system.

In the following section we give some comments on the focus windows from the user's point of view. We summarize the implementation issues and the conclusions in Section 3.

2 Using Focus Windows

We will not describe here the usage of the *Theorema* system. For detailed information on this topic we refer the reader to [3], [10] or to the *Theorema* group's papers (www.theorema.org).

A call for a Theorema prover to work on a proof problem looks like this:

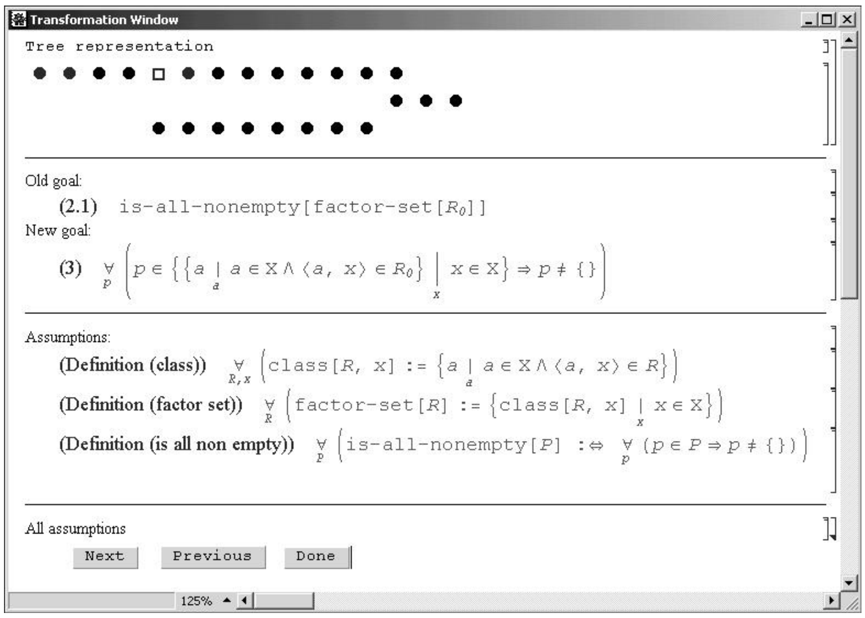
Prove[Lemma["Lm"], using → KnowledgeBase, by → SomeProver,
ProverOptions → {options for the Prover}, **showBy** → **SomeDisplayer**,
DisplayerOptions → {options for the displayer}];

The option that controls the presentation style of a proof is *showBy*. After evaluating the line above and depending on the presentation method the user chooses, either the linear proof is presented or the focused one, where each step of the proof is put into the user's focus [2]. If the user chooses to leave out this option then the linear style is chosen as the default.

When selecting the focus windows presentation style, the user is faced with a step-wise presentation manner. Each step of the proof will be shown to the user in two phases: the attention phase and the transformation phase with a corresponding Attention Window (the formulae produced at the inspected proof step are not yet shown to the user) and a Transformation Window. In the next figure we present an example of a Transformation Window.

Several areas of the window shown in the figure below are common to both Transformation and Attention Window. They are, from top to bottom:
• a "proof tree area" in which the entire proof tree is displayed in a schematic, simplified form,
• a "goal area" where the goal(s) are shown,
• an "assumptions area" in which the relevant assumptions are shown,
• an area (the "all-assumptions area") that presents all the assumptions that are available (not only the relevant ones),

• a "navigation area" that makes navigating in the proof possible,
• debug buttons (only shown if the *FocusDebug* option is set).



Old goal:

(2.1) is-all-nonempty[factor-set[R_0]]

New goal:

(3) $\underset{p}{\forall}\left(p \in \left\{\left\{a \underset{a}{|} a \in X \wedge \langle a, x\rangle \in R_0\right\} \underset{x}{|} x \in X\right\} \Rightarrow p \neq \{\}\right)$

Assumptions:

(Definition (class)) $\underset{R,x}{\forall}\left(\text{class}[R, x] := \left\{a \underset{a}{|} a \in X \wedge \langle a, x\rangle \in R\right\}\right)$

(Definition (factor set)) $\underset{R}{\forall}\left(\text{factor-set}[R] := \left\{\text{class}[R, x] \underset{x}{|} x \in X\right\}\right)$

(Definition (is all non empty)) $\underset{P}{\forall}\left(\text{is-all-nonempty}[P] :\Leftrightarrow \underset{p}{\forall}(p \in P \Rightarrow p \neq \{\})\right)$

In the figure the goal area contains the formulae (2.1) and (3). The latter is the formula that was produced in the proof step presented. Therefore it has the heading 'New Goal'. The assumptions area contains the definitions of the notions that are used at this inference step ('class', 'factor-set' and 'is-all-nonempty'). No new assumptions were derived at this proof step, thus there is no 'New Assumptions' heading. We underline again that the formulae that, in mathematical textbooks, are referred to by labels, are here presented in full form. The user can now easily check the correctness of the inference rule applied. Following the basic philosophy of the focus windows technique, the all-assumption area is closed since the user will normally *not* want to see all assumptions but only the ones that are *relevant* for the current proof step, which are already presented in the assumption area.

Besides being a graphical and schematic representation, the proof tree area has some functionality, too. The nodes of the tree are in one-to-one correspondence with the proof steps of the proof object. The node corresponding to the proof step that is currently seen in the focus window is high-lighted. By clicking any of these nodes, the focus window will display the corresponding proof step information. This allows the user to read the proof in the order he/she prefers. In this paper we emphasize the aspect that this feature is particulary practical for the stage in which new provers are debugged. A prover developer may want to see particular proof steps (final steps, steps where the proof splits into several sub-proofs, etc). The functionality of the proof tree area offers him/her direct access to these nodes, in contrast to the "forward" and "backward" reading sequence – suggested by the prover that generated the proof – and controlled by the 'Next' and 'Previous' buttons. Pressing 'Done' will close the focus window and end this presentation style.

By adding the option *FocusDebug* $\rightarrow$ *True* in the list of DisplayerOptions in the **Prove** call, the user of the above described presentation technique will see at the bottom of the focus window some extra buttons that allows him/her to inspect parts of the underlying proof-object. Since this option is meant for the use of the prover developers, the information shown when pressing these buttons is in raw form.

In addition to what we have described above, the user of the *Theorema* system is offered the possibility to switch from the linear proof presentation style to the focus windows style. This is done in an intuitive way by clicking on a label of a formula in the linear proof. The focus windows presentation will be started and the proof step where the formula attached to the label clicked is produced will be displayed, together with the other information available at this inference step, most importantly: the relevant formulae.

3 Conclusions

We presented a proof presentation technique that emphasizes the content of each inference step of a proof. This technique can, in principle, be applied to both proofs produced by human and automated provers. In the case of proofs that are printed on paper or other media like blackboard, this technique is applied in the mind of the person who studies the proof. When the proofs are available in some processable data structures, the focus windows technique is algorithmically describable. Note that this method is *not* a prove method, but implements a specific way of *presenting* proofs to the users.

We implemented the focus window method in *Mathematica* [11], which is also the language we chose for the implementation of *Theorema*. The implementation did not pose difficulties because of two reasons:

A) *Mathematica*, via its front end, provides convenient programming tools for active objects that, basically, allow to apply the usual Mathematica programming style also for programming man-machine interfaces. We use this facility for attaching certain information to the buttons of the navigation area, to the buttons of the schematic proof tree representation, and to the extra buttons that are shown in the debug-mode, which results in a drastic reduction of the time that the algorithm needs to find a certain proof step in the proof object [7].

B) From the outset, the data structure of *Theorema* proof objects was carefully designed in order to give easy access to the formulae relevant in each proof step [7]. Having a proof-object as a result of an automated prover simplifies the process of proof presentation. This idea is advocated also in [1].

The implementation of the proof presentation technique described above should not be difficult to implement in any existing automated prover, even for systems that do not actually generate proofs automatically but restrict automation to checking proofs generated by humans (like HOL [4], Mizar [5]).The main prerequisite (as suggested by us and by [1]) is that the results of the provers in the system must be formal proof objects that contain sufficient information for post-processing them (in this case, extracting the formulae used and produced in any particular step).

References

1. Y. Bertot, L. Théry. *A Generic Approach To Building User Interfaces for Theorem Provers.* Journal Of Symbolic Computation. Special Issue, Academic Press, Vol. 25, No. 2, February 1998. N.Kajler and M.Monagan eds.
2. B.Buchberger. *Focus Windows Presentation: A New Approach to Presenting Mathematical Proofs (in Automated Theorem Proving Systems). Theorema* Technical Report, 2000–01–30, RISC, http://www.risc.uni–linz.ac.at/people/buchberg/downloads.html
3. B. Buchberger, C. Dupré, T. Jebelean, F. Kriftner, K. Nakagawa, D. Vasaru, W. Windsteiger. *The Theorema Project: A Progress Report.* In: Symbolic Computation and Automated Reasoning (Proceedings of CALCULEMUS 2000, Symposium on the Integration of Symbolic Computation and Mechanized Reasoning, August 6–7, 2000, St. Andrews, Scotland, M. Kerber and M. Kohlhase eds.), A.K. Peters, Natick, Massachusetts, pp. 98–113. ISBN 1–56881–145–4.
4. *The HOL System.* Developed at the University of Cambridge, directed by R. Milner. http://www.cl.cam.ac.uk/Research/HVG/HOL/.
5. *Mizar System.* Developed at the University of Warsaw, directed by A. Trybulec. http://mizar.uwb.edu.pl/system/.
6. *Omega System.* Developed at the University of Saarbrücken, directed by J. Siekmann. http://www.ags.uni–sb.de/ omega/intro.html.
7. F. Piroi, B. Buchberger *Focus Windows: A New Technique for Proof Presentation.* In Lecture Notes in Artificial Intelligence (Proceedings of Calculemus 2002, France, Marseille, July 2002, J. Calmet, B. Benhamou, O. Caproptti, L. Henocque, V. Sorge eds.), Springer Verlag, Berlin, pp.337-341. ISBN 3-540-43865-3.
8. D. Vasaru–Dupré, *Automated Theorem Proving by Integrating Proving, Solving and Computing.* RISC Institute, May 2000, RISC report 00–19. PhD Thesis.
9. F. Wiedijk, *The Fourteen Provers of the World.* 2001, http://www.cs.kun.nl/ freek/notes/index.html
10. W. Windsteiger, *A Set Theory Prover in Theorema: Implementation and Practical Applications*, RISC Institute, May 2001, RISC report 01–03. PhD Thesis.
11. S. Wolfram. *The Mathematica Book*, Wolfram Media and Cambridge University Press, 1999.